Error Analysis of Algorithms for Evaluating Bernstein-Bézier-Type Multivariate Polynomials

J. M. Peña

Abstract. In Computer Aided Geometric Design, the Bernstein-Bézier form is the usual way to store a polynomial defined on a triangle. We perform backward and forward error analysis of the de Casteljau algorithm and of the algorithm proposed by Schumaker and Volk for evaluating such polynomials. The obtained results are also compared with the corresponding results for the bivariate Horner algorithm.

§1. Introduction

In Computer Aided Geometric Design, multivariate polynomials defined on a triangle are usually stored in the Bernstein-Bézier form, and can be evaluated by the de Casteljau algorithm. In [8] a modified Bernstein-Bézier representation of polynomials was introduced, along with an algorithm for its evaluation, which was called the VS algorithm. This algorithm for the evaluation of multivariate polynomials is expressed in terms of nested multiplications, and is more efficient than the de Casteljau algorithm.

Error analysis of the de Casteljau algorithm for univariate polynomials was considered in [2] and [4]. This paper is devoted to backward and forward error analysis of the de Casteljau and VS algorithms for bivariate polynomials. On the other hand, the error analysis of the Horner algorithm for the evaluation of univariate polynomials has been extensively studied in the literature. In fact, backward and forward error analysis of (univariate) Horner's rule was already performed by Wilkinson in [11], pp. 36–37 and 49–50. Other approaches to this problem can be found in [5,9,10,12] (see more references in [3]). An error analysis of the multivariate Horner algorithm has been given in [7]. We also compare our results with those corresponding to the multivariate Horner algorithm.

The paper is organized as follows. Section 2 introduces basic concepts, notation, and auxiliary results. In Section 3 we carry out the mentioned error analysis of the algorithms. Finally, we summarize in Section 4 the main conclusions and the advantages of VS algorithm in this context, taking into account computational cost and forward error analysis.

Curve and Surface Design: Saint-Malo 1999
Pierre-Jean Laurent, Paul Sablonnière, and Larry L. Schumaker (eds.), pp. 315–324.
Copyright ©2000 by Vanderbilt University Press, Nashville, TN.
ISBN 0-8265-1356-5.
All rights of reproduction in any form reserved.

§2. Basic Notations and Auxiliary Results

Let us now introduce some standard notation in error analysis. Given $a \in \mathbb{R}$, the computed element in floating point arithmetic will be denoted by either $\mathrm{fl}(a)$ or by $\hat{a}$. As usual, to investigate the effect of rounding errors when working with floating point arithmetic, we use the model

$$\mathrm{fl}(a \,\mathrm{op}\, b) = (a \,\mathrm{op}\, b)(1+\delta), \quad |\delta| \leq u, \tag{1}$$

with u the unit roundoff and *op* any of the elementary operations $+$, $-$, $\times$, $/$ (see [3, p. 44] for more details). Given $k \in \mathbf{N}_0$ such that $ku < 1$, let us define

$$\gamma_k := \frac{ku}{1-ku}. \tag{2}$$

In our error analysis we shall deal with quantities such that their absolute value is bounded above by γ_k. Following [3], we denote such quantities by θ_k and take into account that by Lemmas 3.3 and 3.1 of [3], the following properties hold:

$$(1+\theta_k)(1+\theta_j) = 1+\theta_{k+j}, \tag{3}$$

and if $\rho_i = \pm 1$, $|\delta_i| \leq u$ $(i = 1, \ldots, k)$ then

$$\prod_{i=1}^{k} (1+\delta_i)^{\rho_i} = 1+\theta_k. \tag{4}$$

In considering the computed solution of a problem, one can try to find the data for which this computed solution is the exact solution. Backward error analysis measures how far these data are from the original data of the problem. So, backward error analysis interprets rounding errors as perturbations in the data. In contrast, forward error analysis measures how far the computed solution is from the exact solution. Therefore, in our evaluation problem, if $\hat{f}(x) = \sum_{i=0}^{n} \bar{c}_i u_i(x)$ is the computed evaluation (instead of the exact evaluation $f(x) = \sum_{i=0}^{n} c_i u_i(x)$), we say that the relative backward error is bounded above by ε if

$$\frac{|\bar{c}_i - c_i|}{|c_i|} \leq \varepsilon, \quad i = 0, \ldots, n.$$

Then we can bound the forward error by

$$|\hat{f}(x) - f(x)| \leq \varepsilon \sum_{i=0}^{n} |c_i u_i(x)|.$$

The number

$$C_u(f(x)) := \sum_{i=0}^{n} |c_i u_i(x)|, \tag{5}$$

measures the stability in the evaluation of a function with respect to perturbations of the coefficients, and is called the condition number for the evaluation of

$f(x)$ with the basis u (see [1,2,4,6,7]). Let us observe that $C_u(f(x))$ depends on the basis u, on the function f, and on the point x. If we assume that the basis is formed by nonnegative functions, (5) can be written as

$$C_u(f(x)) = \sum_{i=0}^{n} |c_i| u_i(x). \tag{6}$$

In conclusion, we can bound the forward error by

$$|\hat{f}(x) - f(x)| \le \varepsilon C_u(f(x)), \tag{7}$$

which is a particular case of the classical formula *(Forward error)* $\le$ *(Backward error)* $\times$ *(Condition number)*.

If we assume that $f(x) \neq 0$ we can also define the relative condition number by

$$c_u(f(x)) := \frac{C_u(f(x))}{|f(x)|}.$$

The following relative forward error bound, analogous to (7), can be derived:

$$\frac{|\hat{f}(x) - f(x)|}{|f(x)|} \le \varepsilon c_u(f(x)).$$

The following result (which was obtained for polynomials in [1]) allows us to compare the condition number of two bases of nonnegative functions in a space when they are related by a nonnegative matrix:

Lemma 1. *Let $\mathcal{U}$ be a finite dimensional vector space of functions defined on $\Omega \subset \mathbb{R}^s$. Let $u = (u_0, \dots, u_n)$, $v = (v_0, \dots, v_n)$ be two bases of nonnegative functions of $\mathcal{U}$ such that*

$$(v_0, \dots, v_n) = (u_0, \dots, u_n)A,$$

where $A = (a_{ij})_{0 \le i,j \le n}$ is a nonnegative matrix. Then $C_u(f(x)) \le C_v(f(x))$ for each function $f \in \mathcal{U}$ evaluated at every $x \in \Omega$.

Proof: Given $f \in \mathcal{U}$, it can be written as

$$f(x) = \sum_{q=0}^{n} c_q v_q(x) = \sum_{i=0}^{n} \left(\sum_{q=0}^{n} (c_q a_{iq}) \right) u_i(x). \tag{8}$$

Then, by (8) and the nonnegativity of u, v and A, we deduce that

$$\begin{aligned} C_u(f(x)) &= \sum_{i=0}^{n} |\sum_{q=0}^{n} (c_q a_{iq})| |u_i(t)| \\ &\le \sum_{i=0}^{n} \sum_{q=0}^{n} |c_q| a_{iq} u_i(x) = \sum_{q=0}^{n} |c_q| \left(\sum_{i=0}^{n} a_{iq} u_i(x) \right) \\ &= \sum_{q=0}^{n} |c_q| \, |v_q(x)| = C_v(f(x)) \end{aligned} \tag{9}$$

for each function $f \in \mathcal{U}$ evaluated at every $x \in \Omega$. □

§3. Backward and Forward Error Analysis of the Algorithms

Given a triangle T with vertices P, Q, R in the plane and a point $X \in T$, let (r, s, t) be the barycentric coordinates of X:

$$X = rP + sQ + tR, \quad r + s + t = 1.$$

Let $\Pi_d(T)$ space of polynomials of total degree d defined on T. Then any polynomial $p \in \Pi_d(T)$ can be written as

$$p(r,s,t) = \sum_{i=0}^{d} \sum_{j=0}^{i} b_{d-i,i-j,j} B^d_{d-i,i-j,j}(r,s,t), \tag{10}$$

where

$$B^d_{i,j,k}(r,s,t) = \frac{d!}{i!j!k!} r^i s^j t^k, \quad i+j+k=d \tag{11}$$

are the Bernstein polynomials. Then (10) is called the Bernstein-Bézier representation of p.

Let us now recall the algorithm of de Casteljau to evaluate the polynomial p at a given (r,s,t). We denote by $b^0_{i,j,k} := b_{i,j,k}$ for all $i+j+k=d$.

Algorithm of de Casteljau

```
for $k = 1$ to $d$
  for $i = 0$ to $d - k$
    for $j = 0$ to $i$
      $b^k_{d-i-k,i-j,j} = r \cdot b^k_{d-i-k+1,i-j,j} + s \cdot b^{k-1}_{d-i-k,i-j+1,j} + t \cdot b^{k-1}_{d-i-k,i-j,j+1}$
    end $j$
  end $i$
end $k$
```

The previous algorithm leads to the evaluation $p(r,s,t) = b^d_{0,0,0}$. It requires $d(d+1)(d+2)/2$ multiplications.

The following result provides backward error analysis of the de Casteljau algorithm for the evaluation of bivariate polynomials.

Theorem 2. *Let us consider the algorithm of de Casteljau in (10) and let us assume that $3du < 1$. Then the computed value $\hat{p}(r,s,t) = \mathrm{fl}(p(r,s,t))$ satisfies*

$$\hat{p}(r,s,t) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{b}_{d-i,i-j,j} B^d_{d-i,i-j,j}(r,s,t), \tag{12}$$

where

$$\frac{|\bar{b}_{i,j,k} - b_{i,j,k}|}{|b_{i,j,k}|} \le \gamma_{3d}. \tag{13}$$

Proof: By (1), for each $k \in \{1, \dots, d\}$

$$\begin{aligned}
&\hat{b}^k_{d-i-k,i-j,j} \\
&\quad = \left[\mathrm{fl}(rb^k_{d-i-k+1,i-j,j}) + \mathrm{fl}(sb^{k-1}_{d-i-k,i-j+1,j} + tb^{k-1}_{d-i-k,i-j,j+1})\right](1+\delta_0) \\
&\quad = \left[(r\hat{b}^k_{d-i-k+1,i-j,j})(1+\delta_1) + (\mathrm{fl}(sb^{k-1}_{d-i-k,i-j+1,j})\right. \\
&\qquad\qquad \left. + \mathrm{fl}(tb^{k-1}_{d-i-k,i-j,j+1}))(1+\delta_2)\right](1+\delta_0) \\
&\quad = \left[(r\hat{b}^k_{d-i-k+1,i-j,j})(1+\delta_1) + ((s\hat{b}^{k-1}_{d-i-k,i-j+1,j})(1+\delta_3)\right. \\
&\qquad\qquad \left. + (t\hat{b}^{k-1}_{d-i-k,i-j,j+1})(1+\delta_4))(1+\delta_2)\right](1+\delta_0),
\end{aligned}$$

where $|\delta_i|$, $i = 0, \dots, 4$, are numbers less than or equal to the unit roundoff u. Then by (4) we can write

$$\begin{aligned}
\hat{b}^k_{d-i-k,i-j,j} = (r\hat{b}^k_{d-i-k+1,i-j,j})(1+\theta_2) + (s\hat{b}^{k-1}_{d-i-k,i-j+1,j})(1+\theta_3) \\
+ (t\hat{b}^{k-1}_{d-i-k,i-j,j+1})(1+\theta_3).
\end{aligned}$$

Iterating the previous argument for $k = d, d-1, \dots, 1$ and comparing the final expression with (10), we deduce (12) and (13). □

By applying the previous result together with formula (7), which relates backward and forward error, we can derive the following corollary on the forward error of the de Casteljau algorithm.

Corollary 3. *Suppose the hypotheses of Theorem 2 hold. Then the computed value $\hat{p}(r,s,t) = \mathrm{fl}(p(r,s,t))$ given by the de Casteljau algorithm satisfies*

$$|p(r,s,t) - \hat{p}(r,s,t)| \le \gamma_{3d} C_B(p(r,s,t)), \tag{14}$$

where B is the Bernstein basis.

A modified Bernstein-Bézier representation of the polynomial p of (10) was considered in [8],

$$p(r,s,t) = \sum_{i=0}^{d}\sum_{j=0}^{i} c_{d-i,i-j,j} r^{d-i} s^{i-j} t^j, \tag{15}$$

where the new coefficients are related with those of (10) by

$$c_{d-i,i-j,j} = \frac{d!}{(d-i)!(i-j)!j!} b_{d-i,i-j,j}, \qquad 0 \le j, i \le d.$$

The following algorithm (which will be called the VS algorithm) was introduced in [8] by Schumaker and Volk to evaluate a polynomial p in the modified Bernstein-Bézier form (15). In contrast with de Casteljau algorithm, this algorithm is expressed in terms of nested multiplications. This version of the algorithm will use the quotients r/t or s/t, assuming that t is bigger than

r and s, in order to avoid divisions by zero or by near-zero values. If r or s is the biggest, it is recommended to adapt the algorithm appropriately.

VS algorithm

```
$A := c_{d,0,0}$
**for** $i = 1$ **to** $d$
  $B := c_{d-i,i,0}$
  **for** $j = 1$ **to** $i$
    $B := B \cdot (s/t) + c_{d-i,i-j,j}$
  **end** $j$
  $A := A \cdot (r/t) + B$
**end** $i$
$p(r,s,t) = A \cdot t^d$.
```

The previous algorithm requires $(d^2 + 5d)/2$ multiplications and two divisions, and so is significantly faster than the de Casteljau algorithm. A backward error analysis of the VS algorithm is performed in the following result.

Theorem 4. *Consider the VS algorithm in (15), and assume that $4du < 1$. Then the computed value $\hat{p}(x) = \mathrm{fl}(p(x))$ satisfies*

$$\hat{p}(x) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{c}_{d-i,i-j,j} r^{d-i} s^{i-j} t^j, \tag{16}$$

where

$$\frac{|\bar{c}_{i,j,k} - c_{i,j,k}|}{|c_{i,j,k}|} \leq \gamma_{4d}. \tag{17}$$

Proof: The VS algorithm consists of a Horner type algorithm which calculates $\frac{p(r,s,t)}{t^d}$ and a last step which multiplies by t^d. From (15) we can write

$$\frac{p(r,s,t)}{t^d} = \sum_{i=0}^{d} \sum_{j=0}^{i} f_{d-i,i-j,j} r^{d-i} s^{i-j} t^j,$$

with $f_{d-i,i-j,j} = \frac{c_{d-i,i-j,j}}{t^d}$. Since by (1) $\mathrm{fl}(r/t) = (r/t)(1+\theta_1)$ and $\mathrm{fl}(s/t) = (s/t)(1+\theta_1)$, we can apply Theorem 3.1 of [7] to the Horner type part of the VS-algorithm, and get

$$\mathrm{fl}\left(\frac{p(r,s,t)}{t^d}\right) = \sum_{i=0}^{d} \sum_{j=0}^{i} \bar{f}_{d-i,i-j,j} r^{d-i} s^{i-j} t^j,$$

where

$$\bar{f}_{d-i,i-j,j} = f_{d-i,i-j,j}(1 + \theta_{3d+1}). \tag{18}$$

Finally, in the last step we have to multiply by t^d, and then by applying $d-1$ times (1), we can obtain (16) with $\bar{c}_{d-i,i-j,j} = \mathrm{fl}(t^d \bar{f}_{d-i,i-j,j}) = t^d \bar{f}_{d-i,i-j,j}(1+\theta_{d-1})$. Thus by (18) and (3)

$$\begin{aligned}\bar{c}_{d-i,i-j,j} &= t^d f_{d-i,i-j,j}(1+\theta_{3d+1})(1+\theta_{d-1}) = t^d f_{d-i,i-j,j}(1+\theta_{4d}) \\ &= c_{d-i,i-j,j}(1+\theta_{4d}). \quad \square\end{aligned}$$

As a consequence of Theorem 4, and applying again formula (7), we can perform a forward error analysis of the VS algorithm:

Corollary 5. *Under the assumptions of Theorem 4, the computed value* $\hat{p}(r,s,t) = \mathrm{fl}(p(r,s,t))$ *of the VS algorithm satisfies*

$$|p(r,s,t) - \hat{p}(r,s,t)| \le \gamma_{4d} C_{\bar{B}}(p(r,s,t)), \tag{19}$$

where $\bar{B}$ *is the basis used in the modified Bernstein-Bézier representation.*

Although in Computer Aided Geometric Design, a bivariate polynomial is usually stored in its Bernstein Bézier form (10) (very close to the modified Bernstein-Bézier representation (15)) we shall compare our error bounds with those obtained by evaluating the polynomial in Taylor form by the bivariate Horner algorithm. Given the triangle T and the polynomial p of total degree d defined on T, let $u = x - x_1$, $v = y - y_1$, where (x_1, y_1) are the cartesian coordinates of a point of T. The Taylor form of p is given by

$$p(u,v) = \sum_{i=0}^{d} \sum_{j=0}^{d-i} a_{i,j} u^i v^j. \tag{20}$$

Bivariate Horner algorithm

```
$p := a_{0,d}$
**for** $i = 1$ **to** $d$
    $A := a_{i,d-i}$
    **for** $j = 1$ **to** $i$
        $A := A \cdot u + a_{i-j,d-i}$
    **end** $j$
**end** $i$
$p = A \cdot v + A.$
```

We observe that the bivariate Horner algorithm requires $(d^2+3d)/2$ multiplications.

The following result, which is a consequence of Corollary 3.2 of [7] for the backward error bound and of formula (7) of the present paper for the forward error bound, provides backward and forward error bounds of the bivariate Horner algorithm.

Theorem 6. *Consider the bivariate Horner algorithm in (20), and assume that $(2d+1)u < 1$. Then the computed value $\hat{p}(u,v) = \mathrm{fl}(p(u,v))$ satisfies*

$$\hat{p}(u,v) = \sum_{i=0}^{d}\sum_{j=0}^{d-i} \bar{a}_{i,j} u^i v^j, \tag{21}$$

where

$$\frac{|\bar{a}_{i,j,k} - a_{i,j,k}|}{|a_{i,j,k}|} \le \gamma_{2d+1}, \tag{22}$$

and satisfies

$$|p(u,v) - \hat{p}(u,v)| \le \gamma_{2d+1} C_M(p(u,v)). \tag{23}$$

§4. Conclusions

As mentioned, in the context of Computer Aided Geometric Design, polynomials are usually stored in the Bernstein-Bézier form (10), which is used by the de Casteljau algorithm. Let us observe that the coefficients of the modified Bernstein-Bézier form (15) used by the VS algorithm are related with those of (10) by

$$c_{d-i,i-j,j} = \frac{d!}{(d-i)!(i-j)!j!} b_{d-i,i-j,j}, \; 0 \le j, i \le d. \tag{24}$$

The conversion from the Bernstein-Bézier form into the modified Bernstein-Bézier form can be done in $(d^2+3d-4)/2$ multiplications. In [8] the algorithm composed of the conversion from the Bernstein-Bézier form into the modified Bernstein-Bézier form followed by the VS algorithm was called the VSC algorithm. The number of multiplications and divisions required by the VSC algorithm is $d^2 + 4d$.

We have seen that the VS algorithm is considerably more efficient than the de Casteljau algorithm. On the other hand, the bivariate Horner algorithm is also very efficient and has the lowest backward error bound, as one can deduce from the results in the previous section. However the behaviour with respect to the forward errors is different as we shall now show.

Given the bivariate Bernstein basis B defined on a triangle (see (11)), since the barycentric coordinates satisfy $r+s+t=1$ we can change the variables in the form $(r,s,t) = (u,v,1-u-v)$ and obtain the following expression of the functions in B:

$$B_{i,j}^d(u,v) = \binom{d}{i,j} u^i v^j (1-u-v)^{d-i-j}, \quad u,v \in [0,1],$$

where

$$\binom{d}{i,j} := \frac{d!}{(d-i-j)!i!j!}.$$

With the same change of variables, the functions of the basis $\bar{B}$ (used in the modified Bernstein form) can be written as

$$\bar{B}^d_{i,j}(u,v) = u^i v^j (1-u-v)^{d-i-j}, \quad u,v \in [0,1].$$

Thus, these functions are related with those of B by

$$B^d_{i,j}(u,v) = \binom{d}{i,j} \bar{B}^d_{i,j}(u,v).$$

We see that the basis functions of B are obtained from those of $\bar{B}$ by a positive scaling. Then, using the condition number of (5), it is easy to prove that

$$C_{\bar{B}}(p(u,v)) = C_B(p(u,v)), \quad \forall p, \forall u,v \in [0,1]. \tag{25}$$

On the other hand, the Taylor form uses the power basis M formed by the functions $u^i v^j$, $0 \le i \le d$, $0 \le j \le d-i$, $u,v \in [0,1]$. By formula (100) of [2], the functions of the power basis M can be expressed as

$$u^i v^j = \sum_{k=i}^{n-j} \sum_{l=j}^{n-k} \frac{\binom{k}{i}\binom{l}{j}}{\binom{d}{i,j}} B^d_{kl}(u,v).$$

Since the coefficients are positive and the basis functions are nonnegative, we can deduce from Lemma 1 that $C_B(p(u,v)) \le C_M(p(u,v))$. Therefore, by (25), we conclude for every polynomial $p(u,v)$,

$$C_{\bar{B}}(p(u,v)) = C_B(p(u,v)) \le C_M(p(u,v)), \quad u,v \in [0,1]. \tag{26}$$

In consequence, although the bivariate Horner algorithm provided lower backward error bounds than de Casteljau and VS algorithms, formula (26) shows that these algorithms use better conditioned bases, and this fact reduces their corresponding forward error bounds.

In conclusion, the VS algorithm has more advantages than de Casteljau or Horner algorithms in this context due to the following properties. First, it uses a basis very close to the Bernstein basis, which is more appropriate in the field of Computer Aided Geometric Design than the power basis. Secondly, the bases used by de Casteljau algorithm and the VS algorithm are also better conditioned than the power basis in the considered domain, this last property being convenient from the point of view of forward error analysis. Finally, the VS algorithm has a higher efficiency than the de Casteljau algorithm.

Acknowledgments. This research has been partially supported by the spanish Research Grant DGICYT PB96-0730.

References

1. Farouki, R. T. and T. N. T. Goodman, On the optimal stability of Bernstein basis, Math. Comp. **65** (1996), 1553–1566.
2. Farouki, R. T. and V. T. Rajan, On the numerical condition of polynomials in Bernstein form, Comput. Aided Geom. Design **4** (1987), 191–216.
3. Higham, N. J., *Accuracy and stability of numerical algorithms*, SIAM, Philadelphia, 1996.
4. Mainar, E. and J. M. Peña, Error analysis of corner cutting algorithms, Numerical Algorithms, to appear.
5. Olver, F. W. J., Error bounds for polynomial evaluation and complex arithmetic, IMA J. Numer. Anal. **2** (1982), 377–406.
6. Peña, J. M., B-splines and optimal stability, Math. Comp. **66** (1997), 1555–1560.
7. Peña J. M. and T. Sauer, On the multivariate Horner scheme. SIAM J. Numer. Anal., to appear.
8. Schumaker, L. L. and W.Volk, Efficient evaluation of multivariate polynomials, Comput. Aided Geom. Design **3** (1986), 149–154.
9. Stewart, G. W., Error analysis of the algorithm for shifting the zeros of a polynomial by synthetic division, Math. Comp. **25** (1971), 135–139.
10. Tsao, N. K., Error analysis of splitting algorithms for polynomials, Numer. Math. **32** (1979), 409–421.
11. Wilkinson, J. H., *Rounding errors in algebraic processes*, Notes on Applied Science 32, Her Majesty's Stationery Office, London, 1963.
12. Wozniakowski, H., Rounding error analysis for the evaluation of a polynomial and some of its derivatives, SIAM J. Numer. Anal. **11** (1974), 780–787.

J. M. Peña
Departamento de Matemática Aplicada
Universidad de Zaragoza
50006 Zaragoza, Spain
`jmpena@posta.unizar.es`